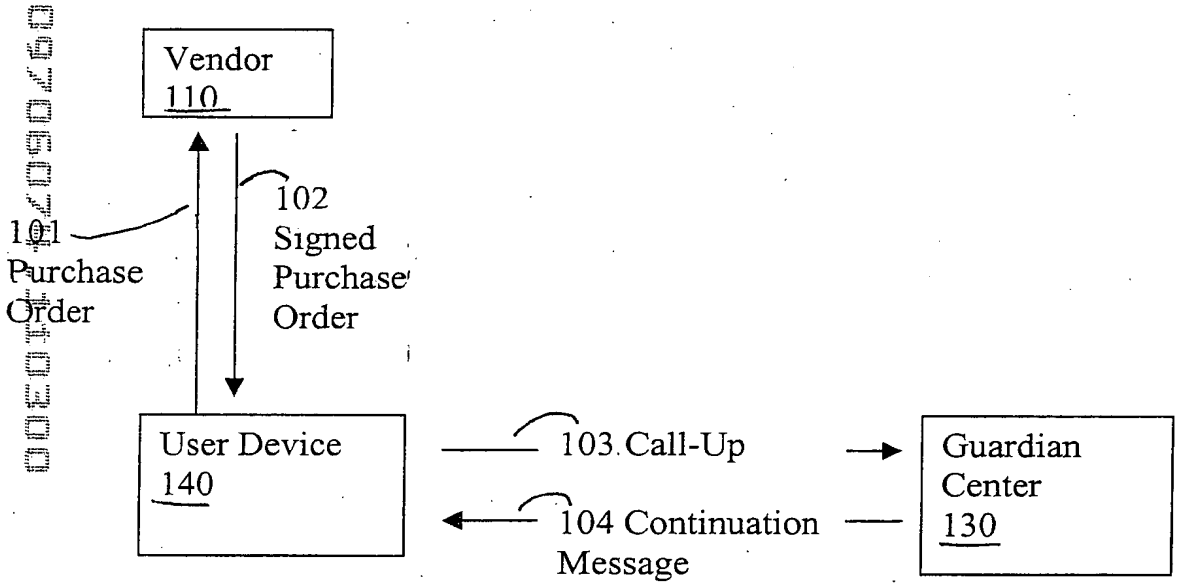

Vendor
110
101
Purchase
Order
102
Signed
Purchase
Order
User Device
140
103. Call-Up
104 Continuation
Message
Guardian
Center
130

Fig. 1

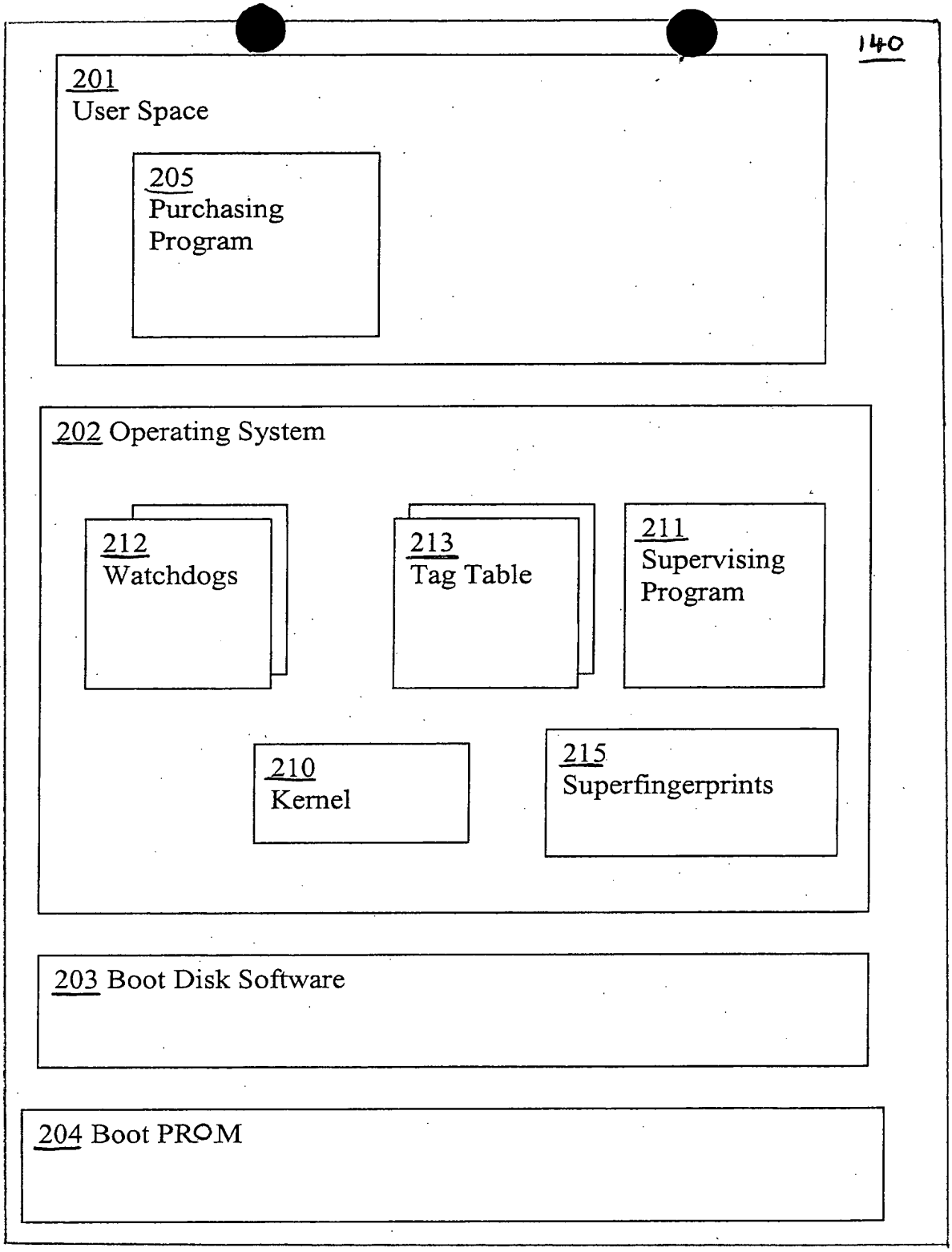

140
201
User Space
205
Purchasing
Program
202 Operating System
212
Watchdogs
213
Tag Table
211
Supervising
Program
210
Kernel
215
Superfingerprints
203 Boot Disk Software
204 Boot PROM

Fig. 2

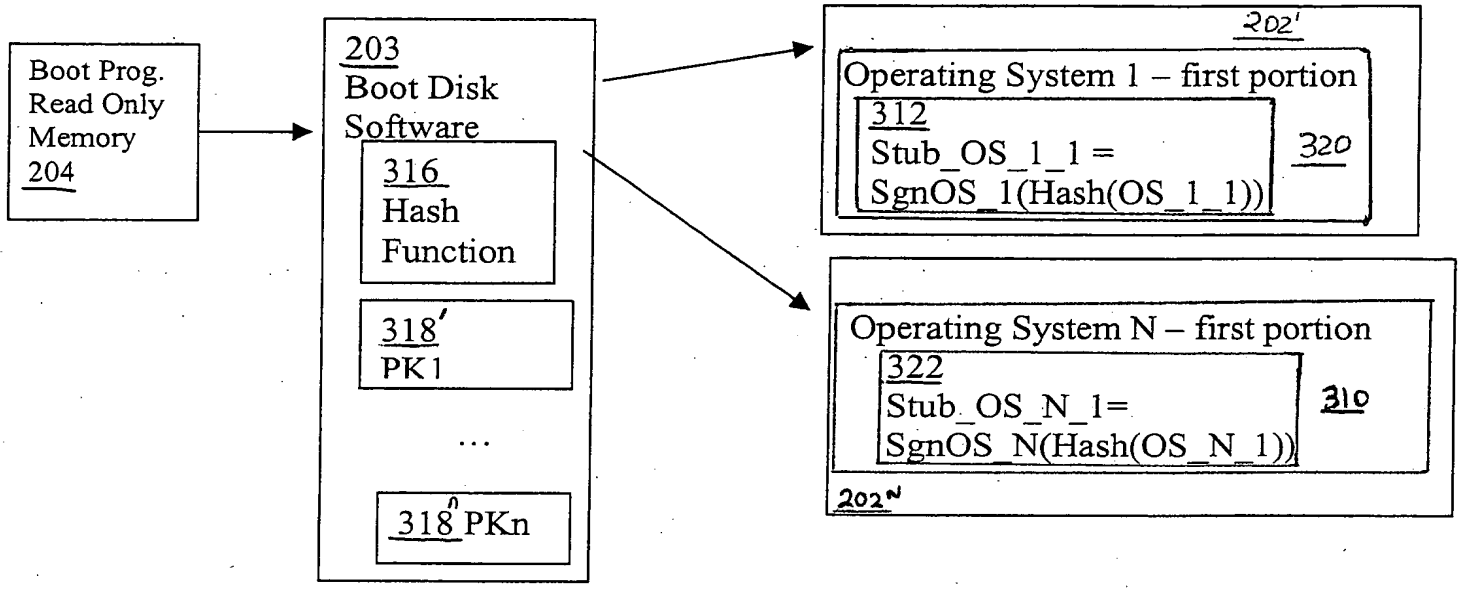


FIG. 3

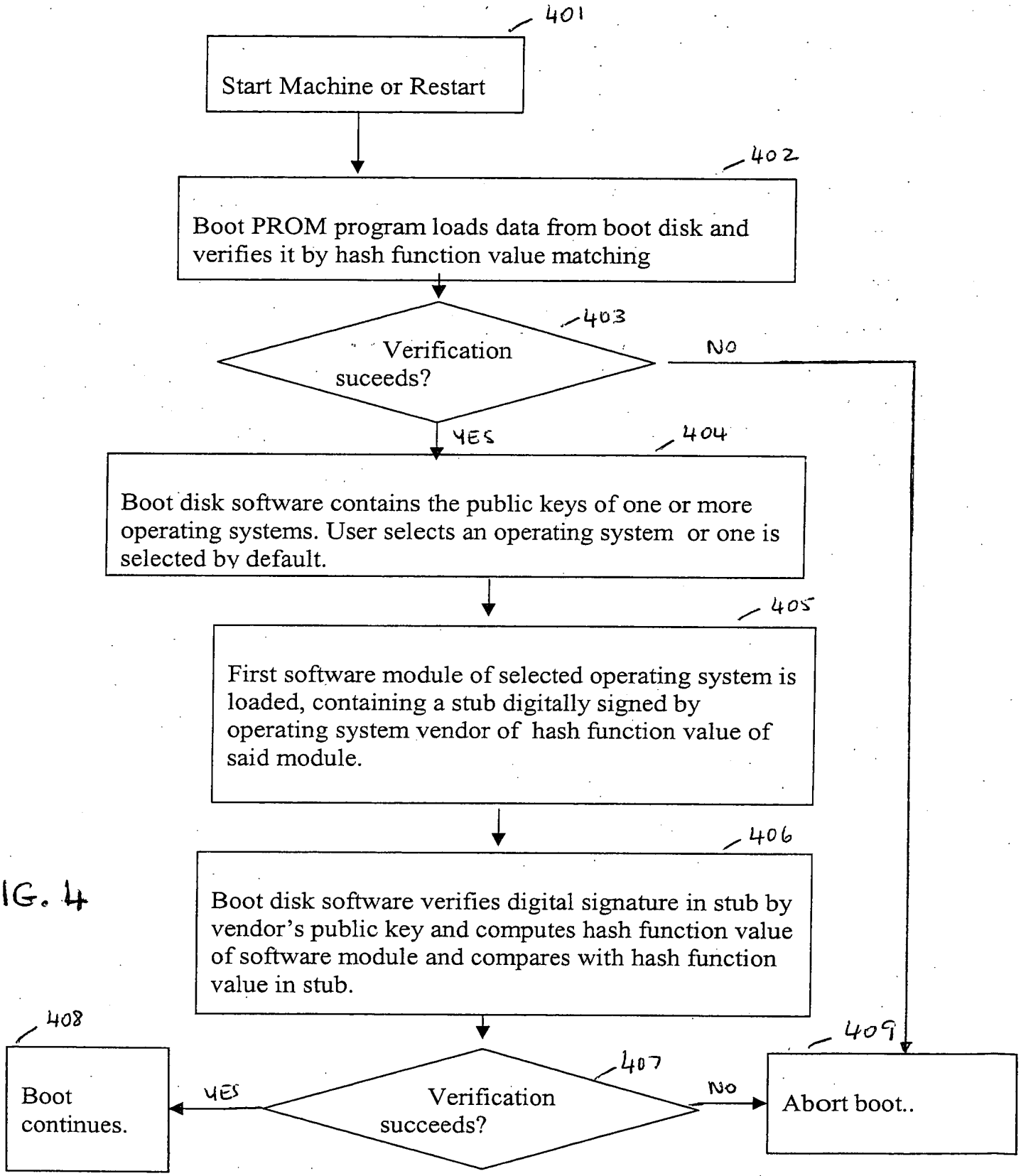

401
Start Machine or Restart
402
Boot PROM program loads data from boot disk and verifies it by hash function value matching
403
Verification suceeds?
NO
YES
404
Boot disk software contains the public keys of one or more operating systems. User selects an operating system or one is selected by default.
405
First software module of selected operating system is loaded, containing a stub digitally signed by operating system vendor of hash function value of said module.
406
Boot disk software verifies digital signature in stub by vendor's public key and computes hash function value of software module and compares with hash function value in stub.
407
Verification succeeds?
YES
NO
408
Boot continues.
409
Abort boot..

FIG. 4

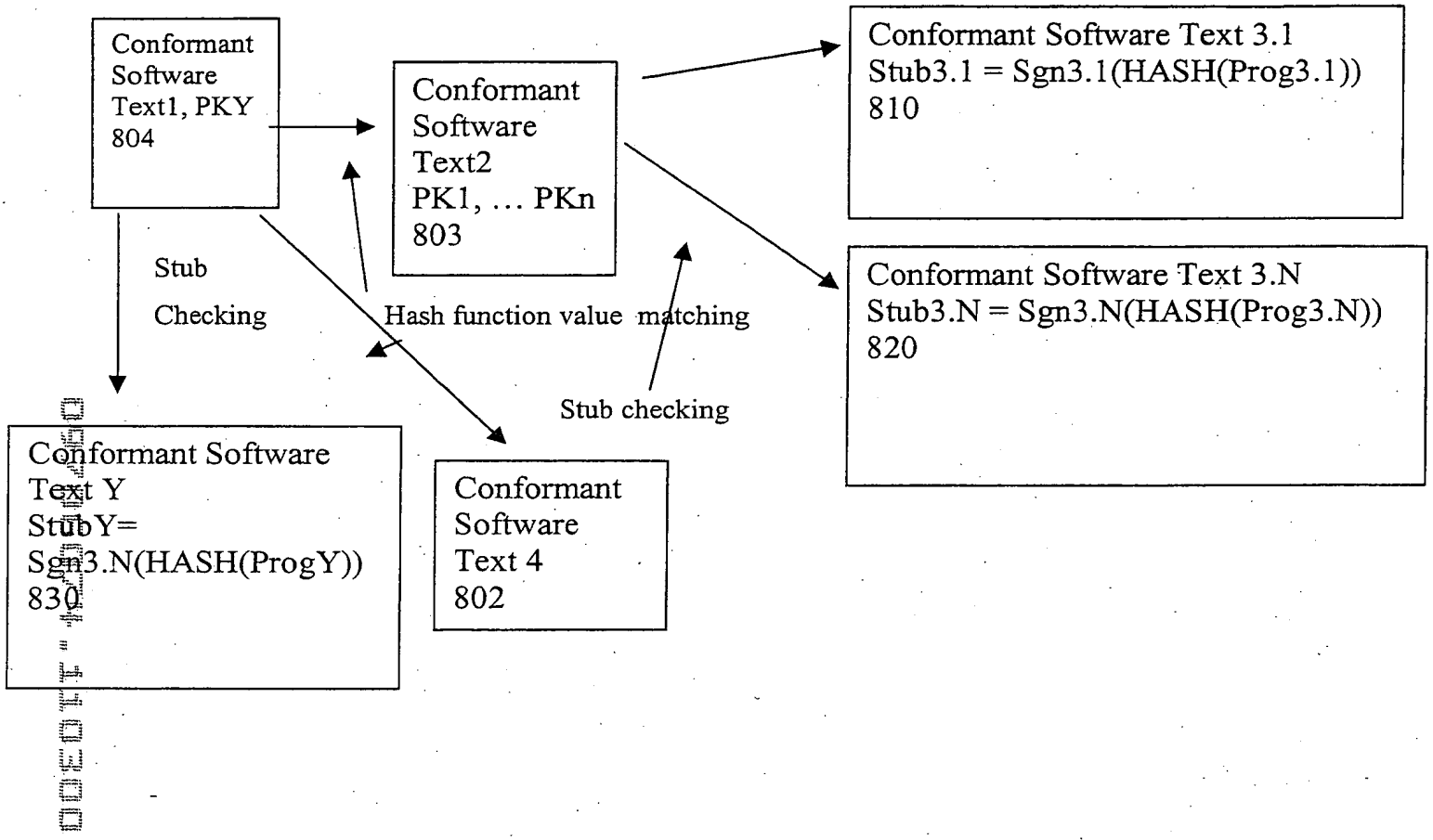

Conformant Software Text1, PKY 804
Conformant Software Text2 PK1, … PKn 803
Conformant Software Text 3.1 Stub3.1 = Sgn3.1(HASH(Prog3.1)) 810
Conformant Software Text 3.N Stub3.N = Sgn3.N(HASH(Prog3.N)) 820
Stub Checking
Hash function value matching
Stub checking
Conformant Software Text Y StubY= Sgn3.N(HASH(ProgY)) 830
Conformant Software Text 4 802

FIG. 5

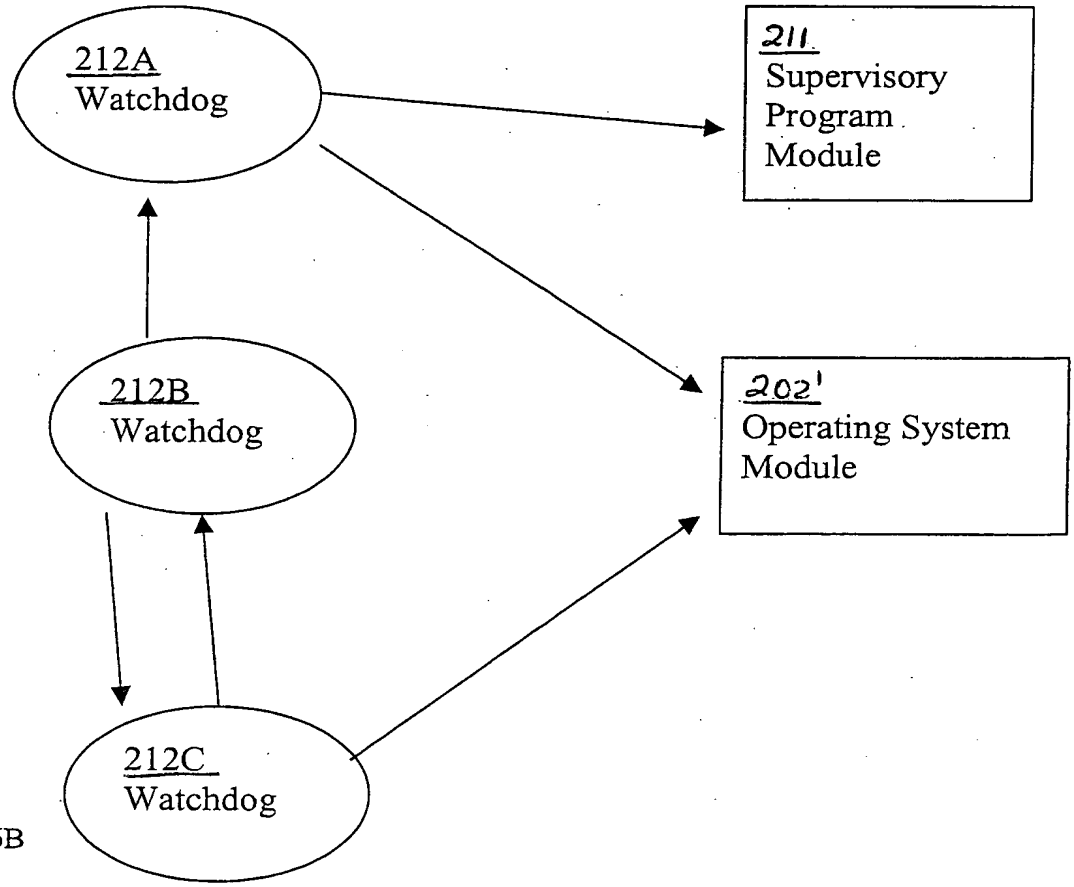


Fig. 6B

Fig. 6A

530
Watchdog Structure

Optional need-to-check test 532
Addresses to be checked 534
Hash functions 536
Hash function values 538
Watchdog actions 540

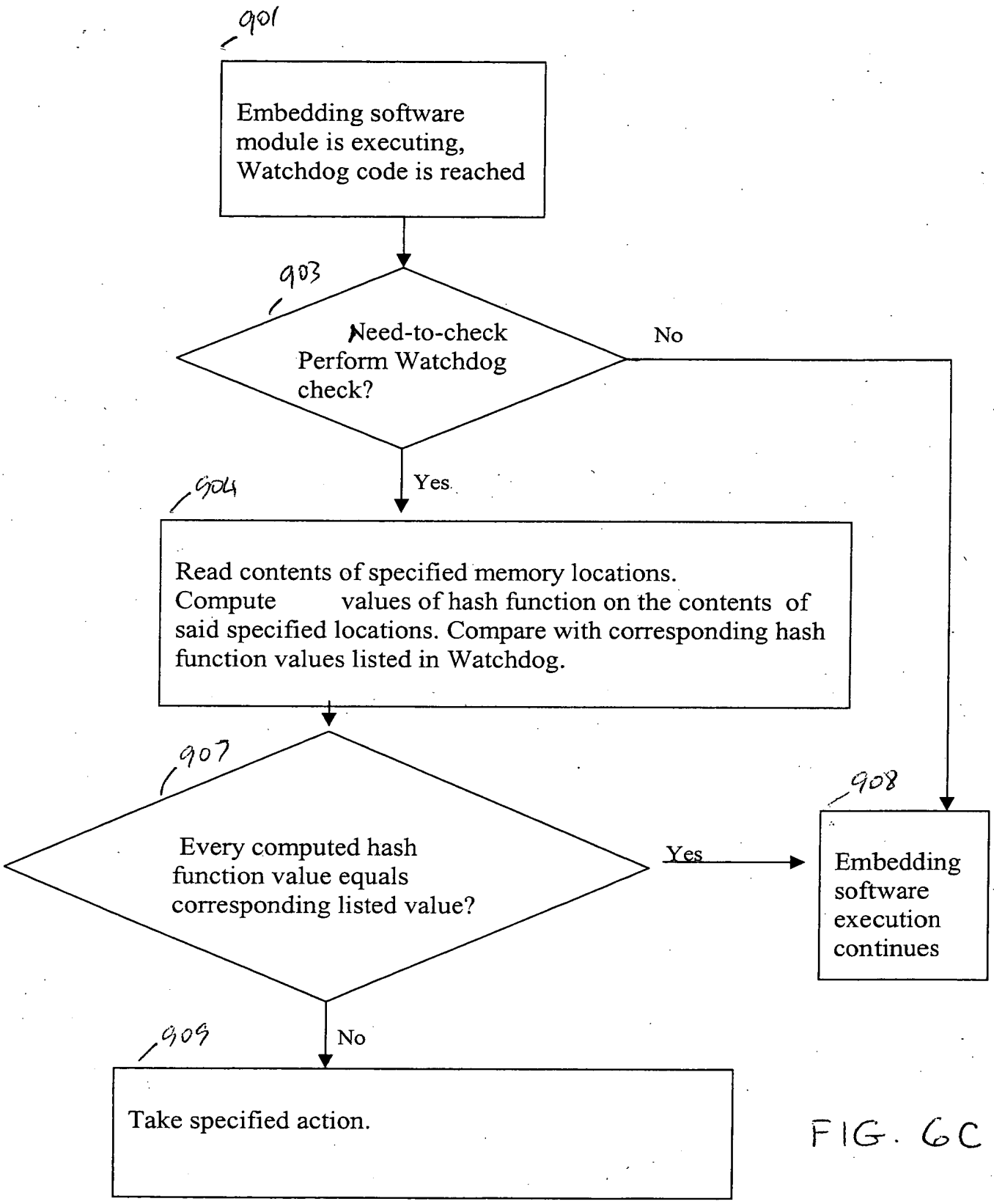

901
Embedding software module is executing, Watchdog code is reached
903
Need-to-check Perform Watchdog check?
No
Yes
904
Read contents of specified memory locations. Compute values of hash function on the contents of said specified locations. Compare with corresponding hash function values listed in Watchdog.
907
Every computed hash function value equals corresponding listed value?
Yes
908
Embedding software execution continues
No
909
Take specified action.

FIG. 6C

Fig. 6D

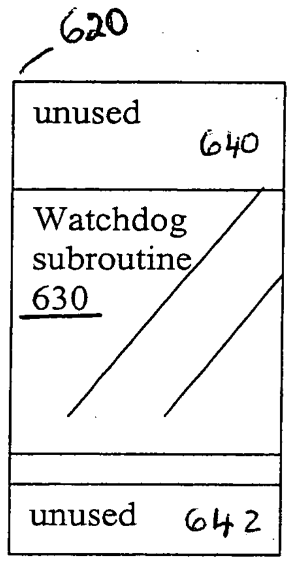
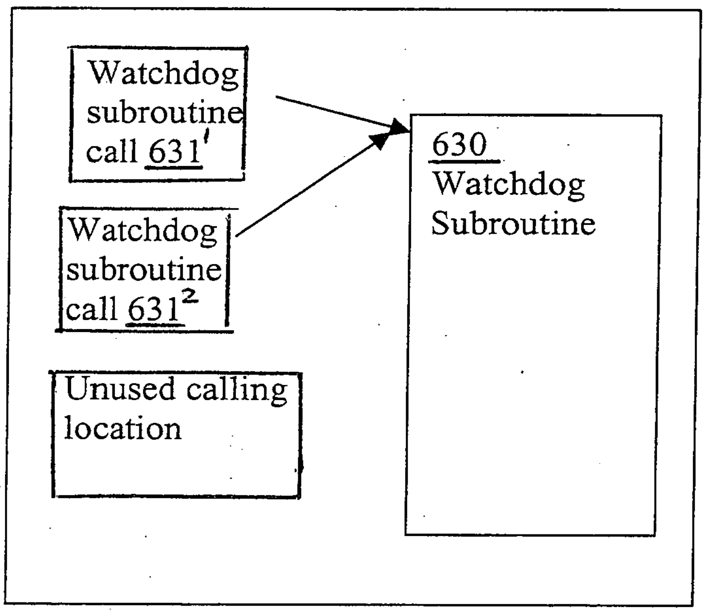


Fig. 6E



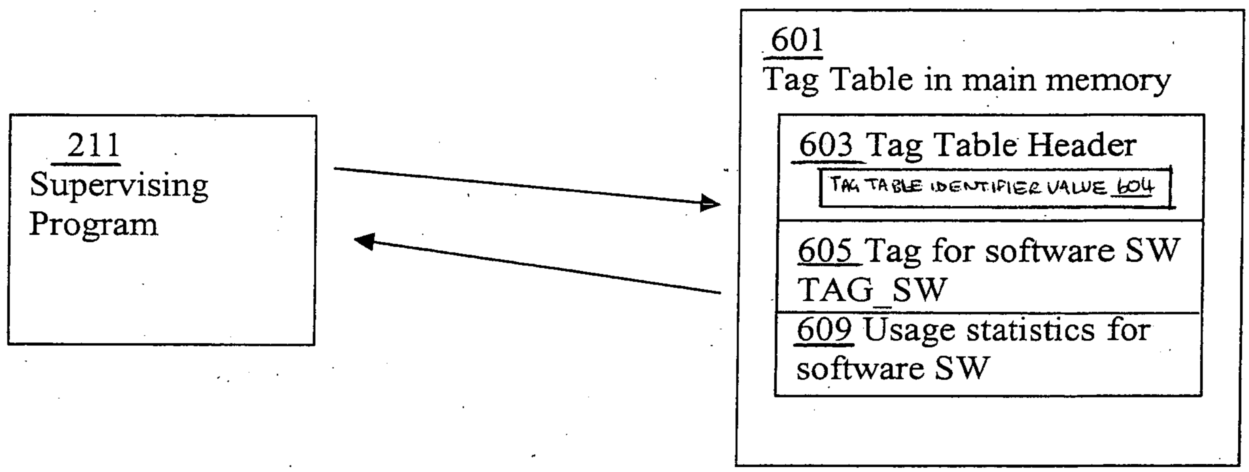
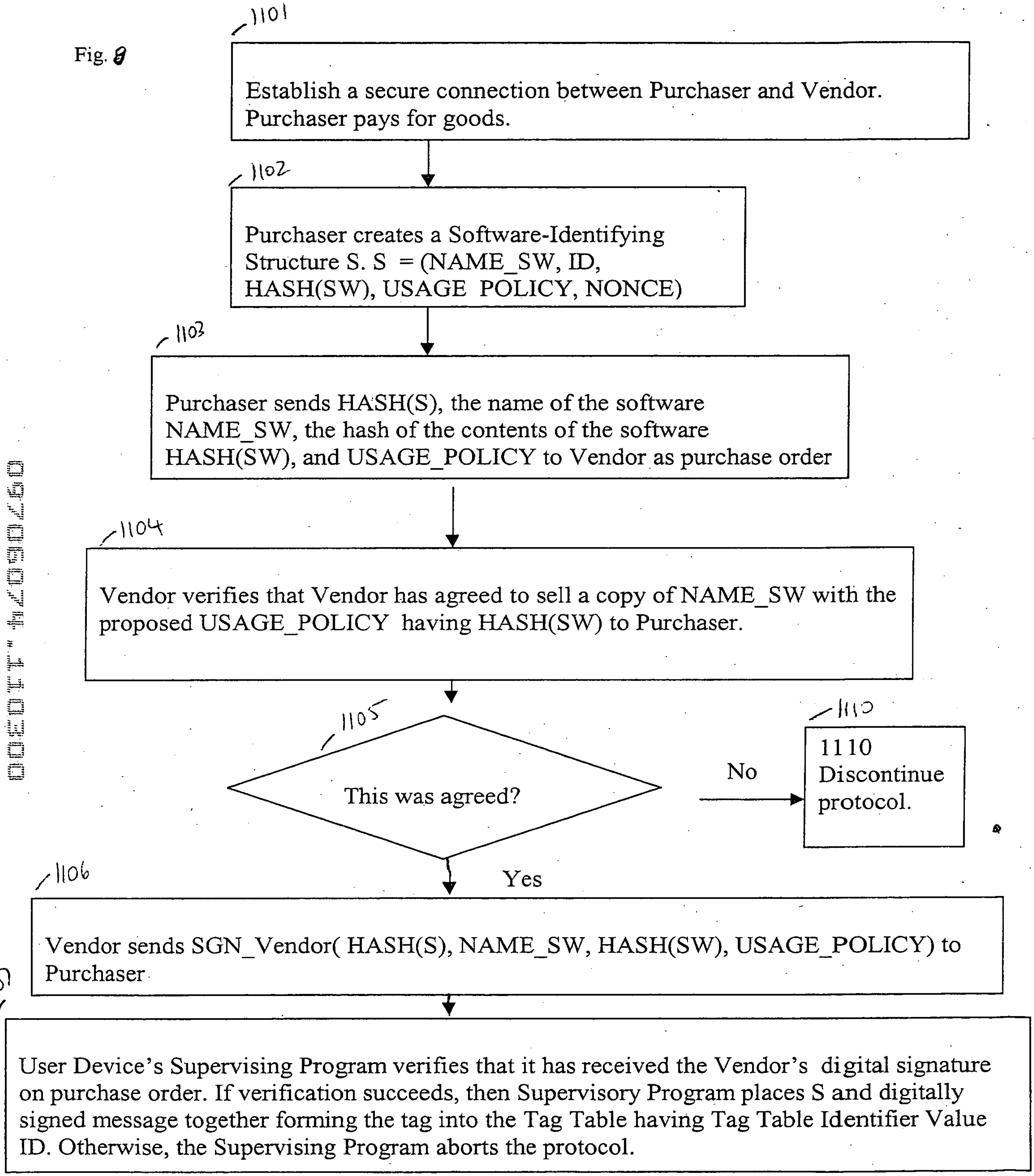

211
Supervising Program
601
Tag Table in main memory
603 Tag Table Header
TAG TABLE IDENTIFIER VALUE 604
605 Tag for software SW
TAG_SW
609 Usage statistics for software SW

Fig. 7

Fig. 8


1101
Establish a secure connection between Purchaser and Vendor. Purchaser pays for goods.
1102
Purchaser creates a Software-Identifying Structure S. S = (NAME_SW, ID, HASH(SW), USAGE POLICY, NONCE)
1103
Purchaser sends HASH(S), the name of the software NAME_SW, the hash of the contents of the software HASH(SW), and USAGE_POLICY to Vendor as purchase order
1104
Vendor verifies that Vendor has agreed to sell a copy of NAME_SW with the proposed USAGE_POLICY having HASH(SW) to Purchaser.
1105
This was agreed?
No
1110
1110 Discontinue protocol.
Yes
1106
Vendor sends SGN_Vendor(HASH(S), NAME_SW, HASH(SW), USAGE_POLICY) to Purchaser
1107
User Device's Supervising Program verifies that it has received the Vendor's digital signature on purchase order. If verification succeeds, then Supervisory Program places S and digitally signed message together forming the tag into the Tag Table having Tag Table Identifier Value ID. Otherwise, the Supervising Program aborts the protocol.

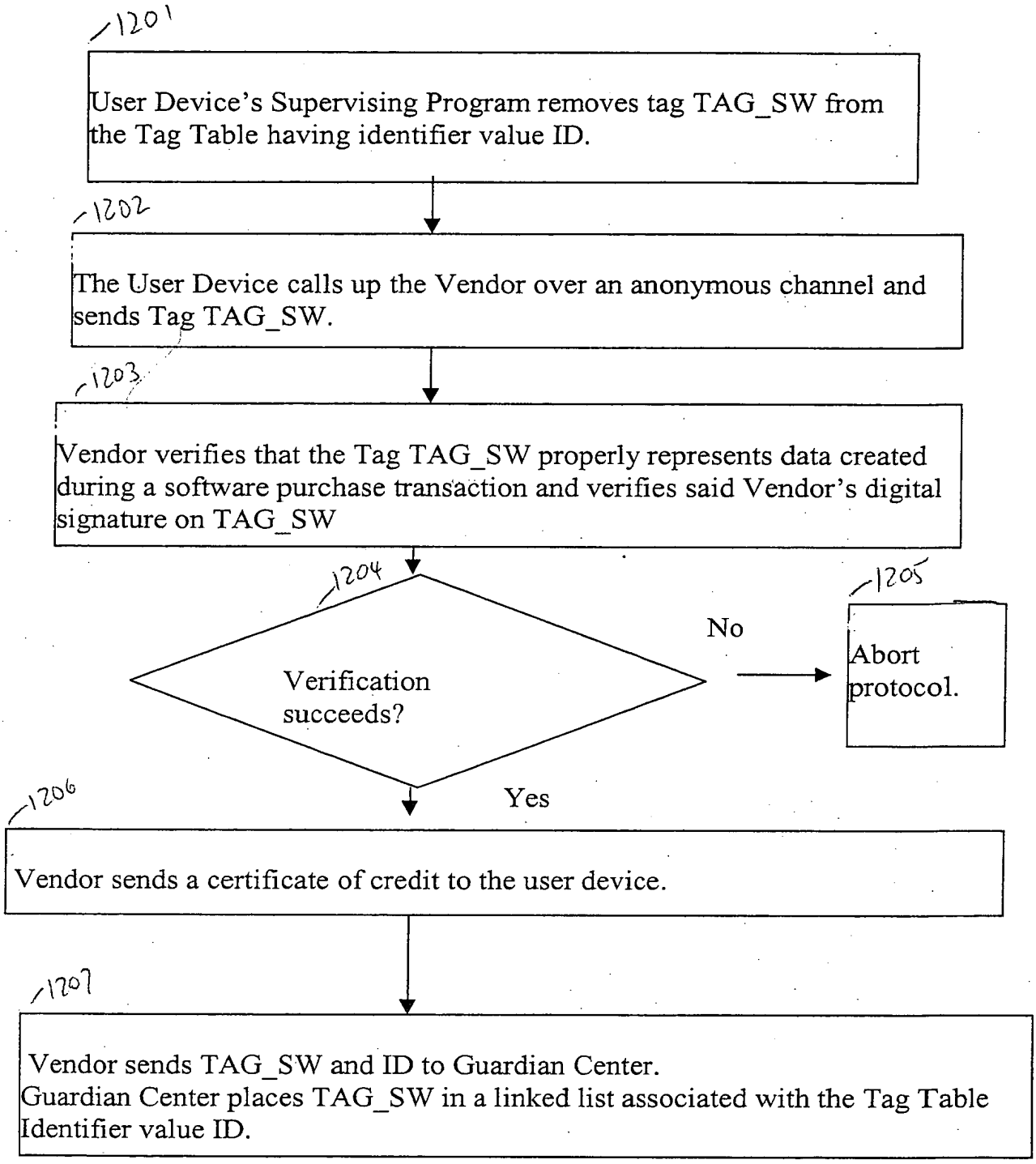

1201
User Device's Supervising Program removes tag TAG_SW from the Tag Table having identifier value ID.
1202
The User Device calls up the Vendor over an anonymous channel and sends Tag TAG_SW.
1203
Vendor verifies that the Tag TAG_SW properly represents data created during a software purchase transaction and verifies said Vendor's digital signature on TAG_SW
1204
Verification succeeds?
No
1205
Abort protocol.
Yes
1206
Vendor sends a certificate of credit to the user device.
1207
Vendor sends TAG_SW and ID to Guardian Center.
Guardian Center places TAG_SW in a linked list associated with the Tag Table Identifier value ID.

FIG. 9

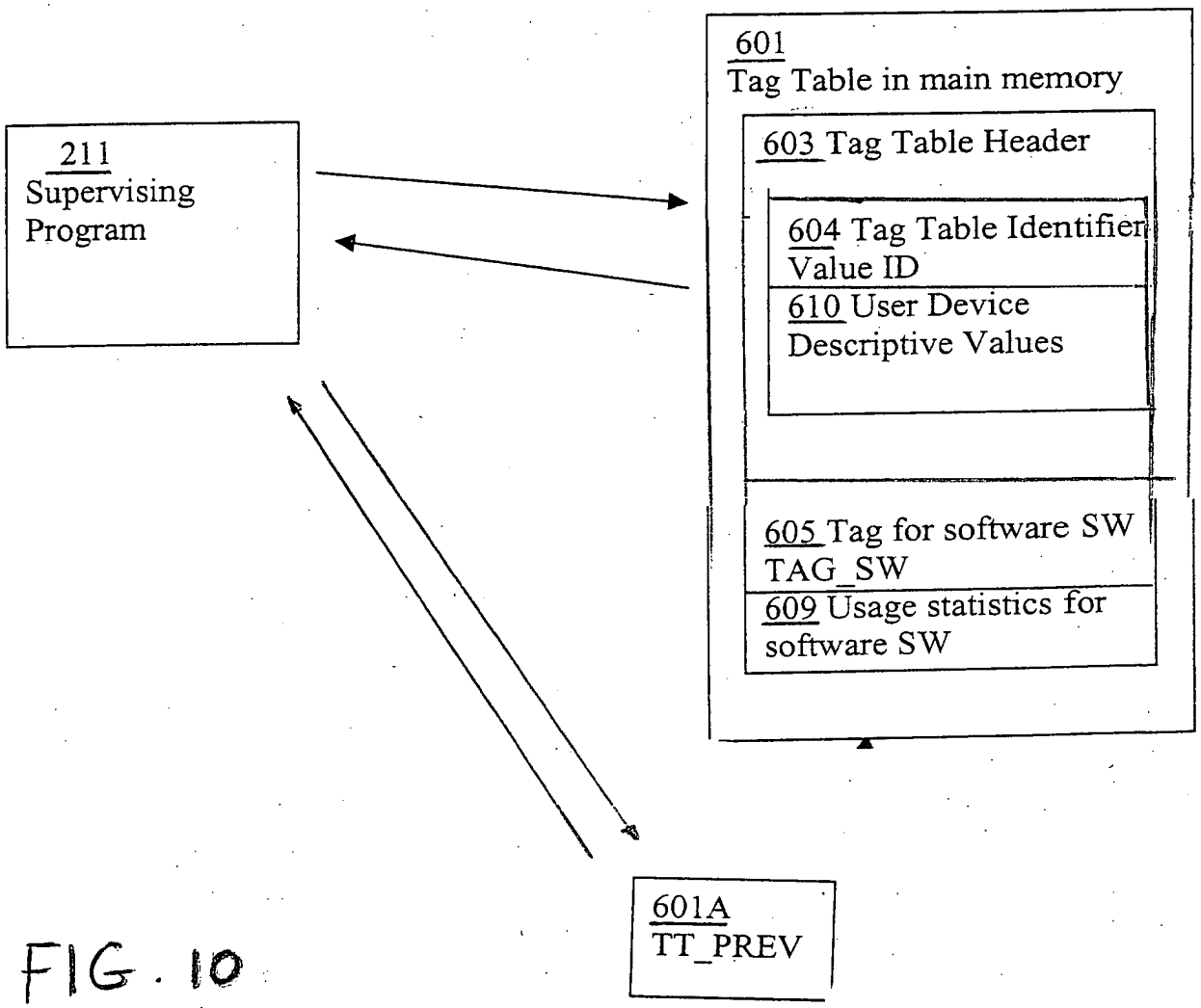


FIG. 10

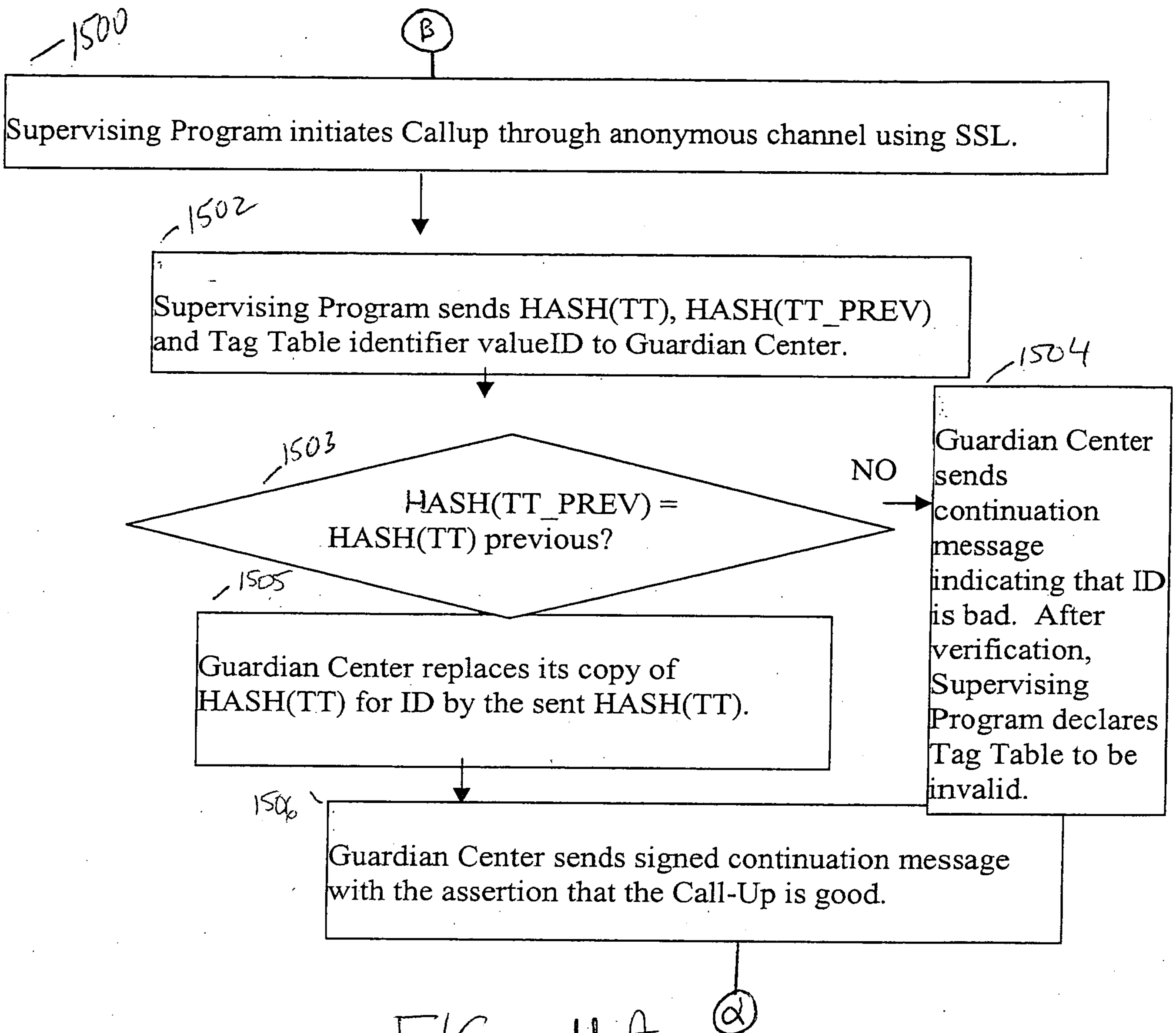

1500
B
Supervising Program initiates Callup through anonymous channel using SSL.
1502
Supervising Program sends HASH(TT), HASH(TT_PREV) and Tag Table identifier valueID to Guardian Center.
1503
HASH(TT_PREV) = HASH(TT) previous?
NO
1504
Guardian Center sends continuation message indicating that ID is bad. After verification, Supervising Program declares Tag Table to be invalid.
1505
Guardian Center replaces its copy of HASH(TT) for ID by the sent HASH(TT).
1506
Guardian Center sends signed continuation message with the assertion that the Call-Up is good.

FIG. 11A

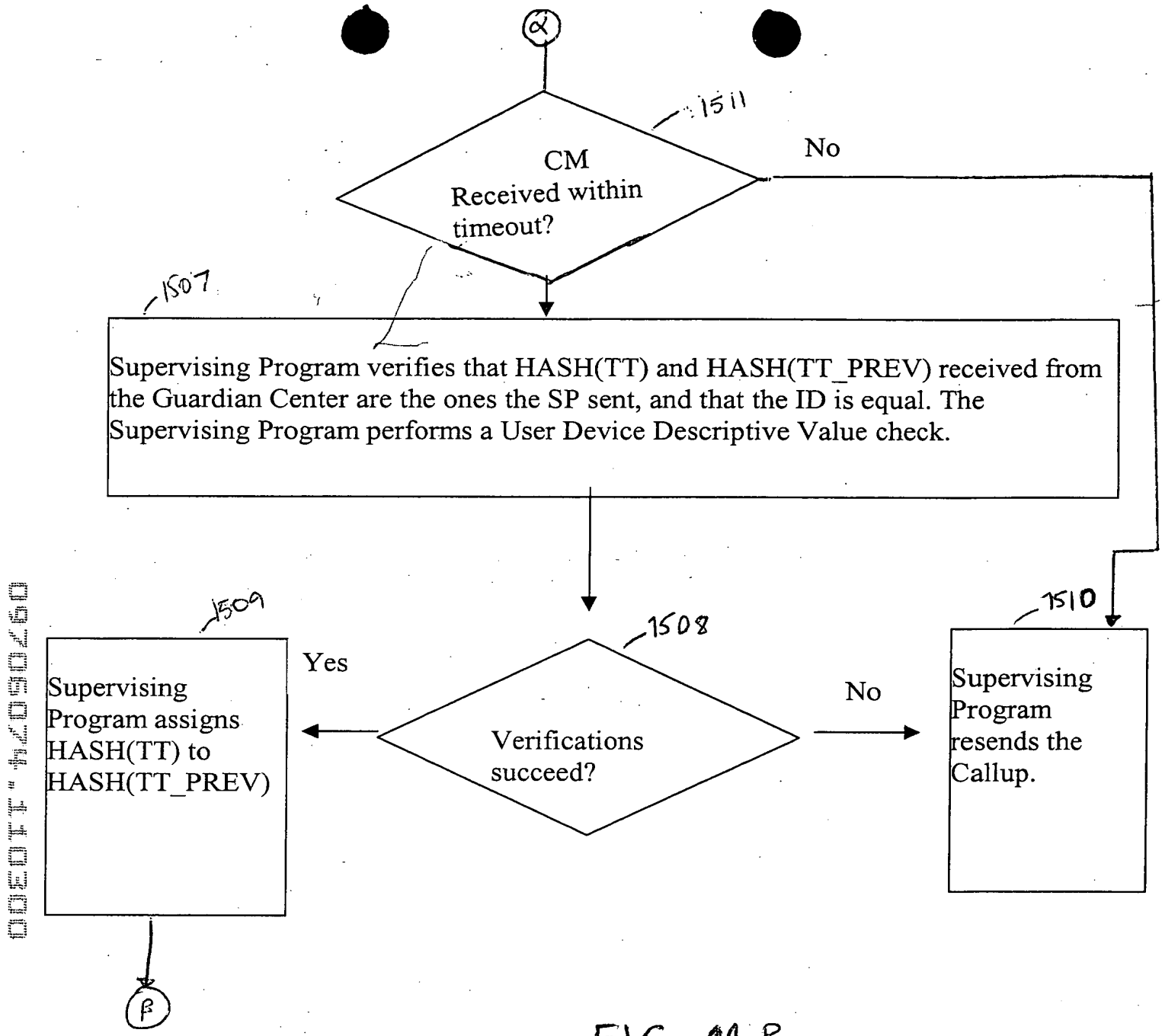

α
1511
CM
Received within
timeout?
No
1507
Supervising Program verifies that HASH(TT) and HASH(TT_PREV) received from the Guardian Center are the ones the SP sent, and that the ID is equal. The Supervising Program performs a User Device Descriptive Value check.
1509
1508
1510
Supervising
Program assigns
HASH(TT) to
HASH(TT_PREV)
Yes
Verifications
succeed?
No
Supervising
Program
resends the
Callup.
β

FIG. 11 B

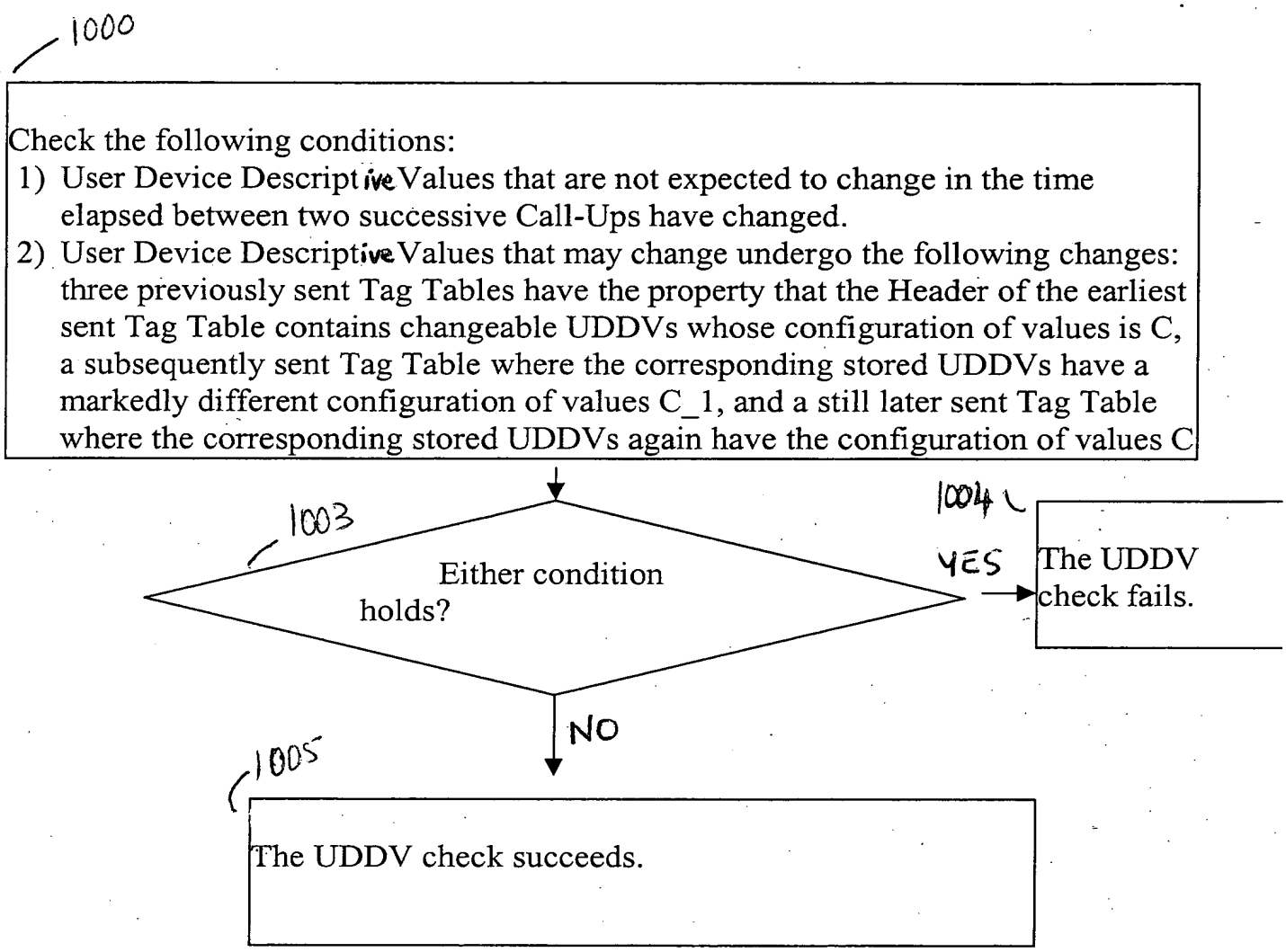

1000
Check the following conditions:
1) User Device Descriptive Values that are not expected to change in the time elapsed between two successive Call-Ups have changed.
2) User Device Descriptive Values that may change undergo the following changes: three previously sent Tag Tables have the property that the Header of the earliest sent Tag Table contains changeable UDDVs whose configuration of values is C, a subsequently sent Tag Table where the corresponding stored UDDVs have a markedly different configuration of values C_1, and a still later sent Tag Table where the corresponding stored UDDVs again have the configuration of values C
1003
Either condition holds?
1004
YES
The UDDV check fails.
NO
1005
The UDDV check succeeds.

FIG. 12

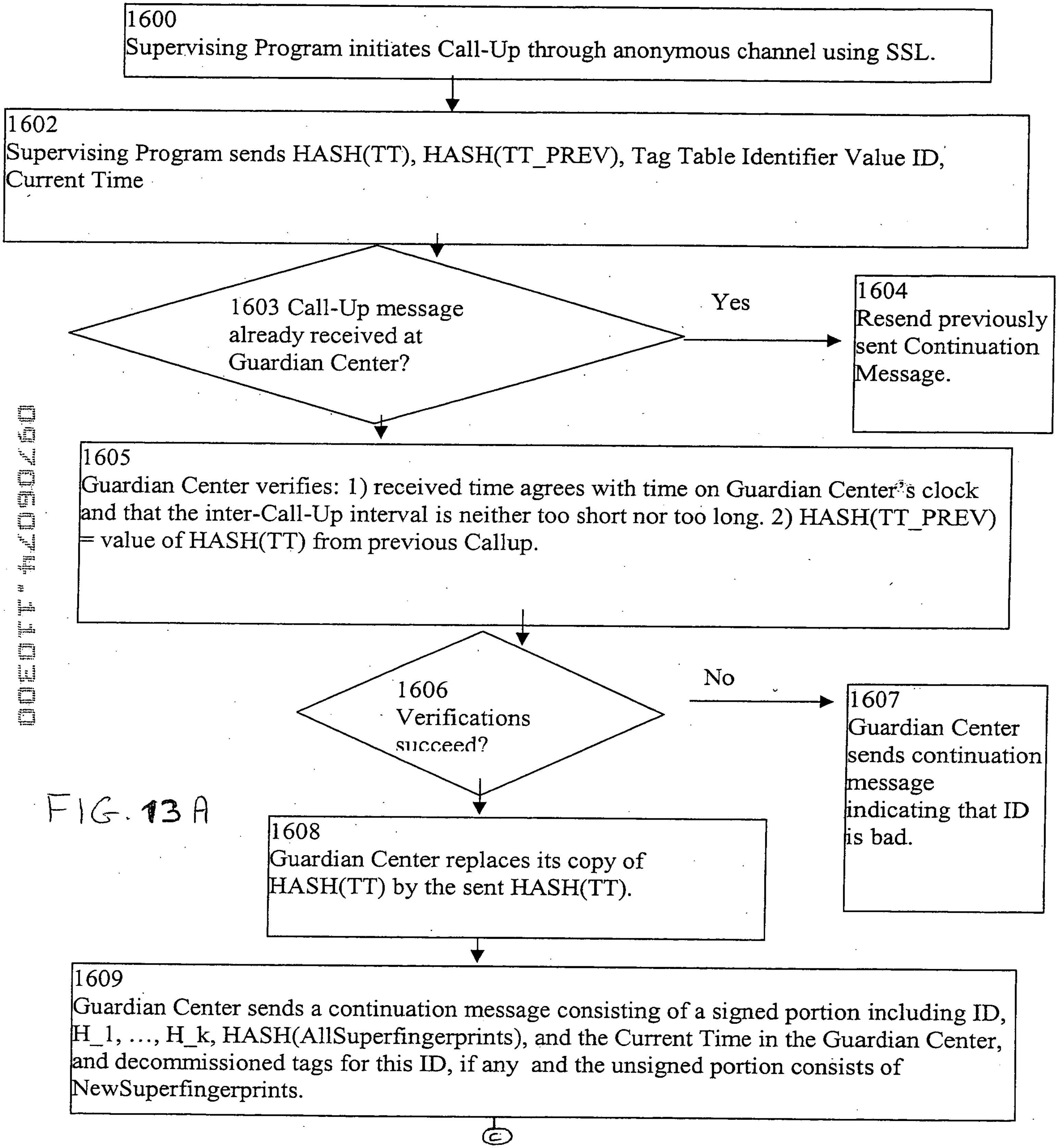

1600
Supervising Program initiates Call-Up through anonymous channel using SSL.
1602
Supervising Program sends HASH(TT), HASH(TT_PREV), Tag Table Identifier Value ID, Current Time
1603 Call-Up message already received at Guardian Center?
Yes
1604
Resend previously sent Continuation Message.
1605
Guardian Center verifies: 1) received time agrees with time on Guardian Center's clock and that the inter-Call-Up interval is neither too short nor too long. 2) HASH(TT_PREV) = value of HASH(TT) from previous Callup.
1606
Verifications succeed?
No
1607
Guardian Center sends continuation message indicating that ID is bad.
1608
Guardian Center replaces its copy of HASH(TT) by the sent HASH(TT).
1609
Guardian Center sends a continuation message consisting of a signed portion including ID, H_1, …, H_k, HASH(AllSuperfingerprints), and the Current Time in the Guardian Center, and decommissioned tags for this ID, if any and the unsigned portion consists of NewSuperfingerprints.
C

FIG. 13A

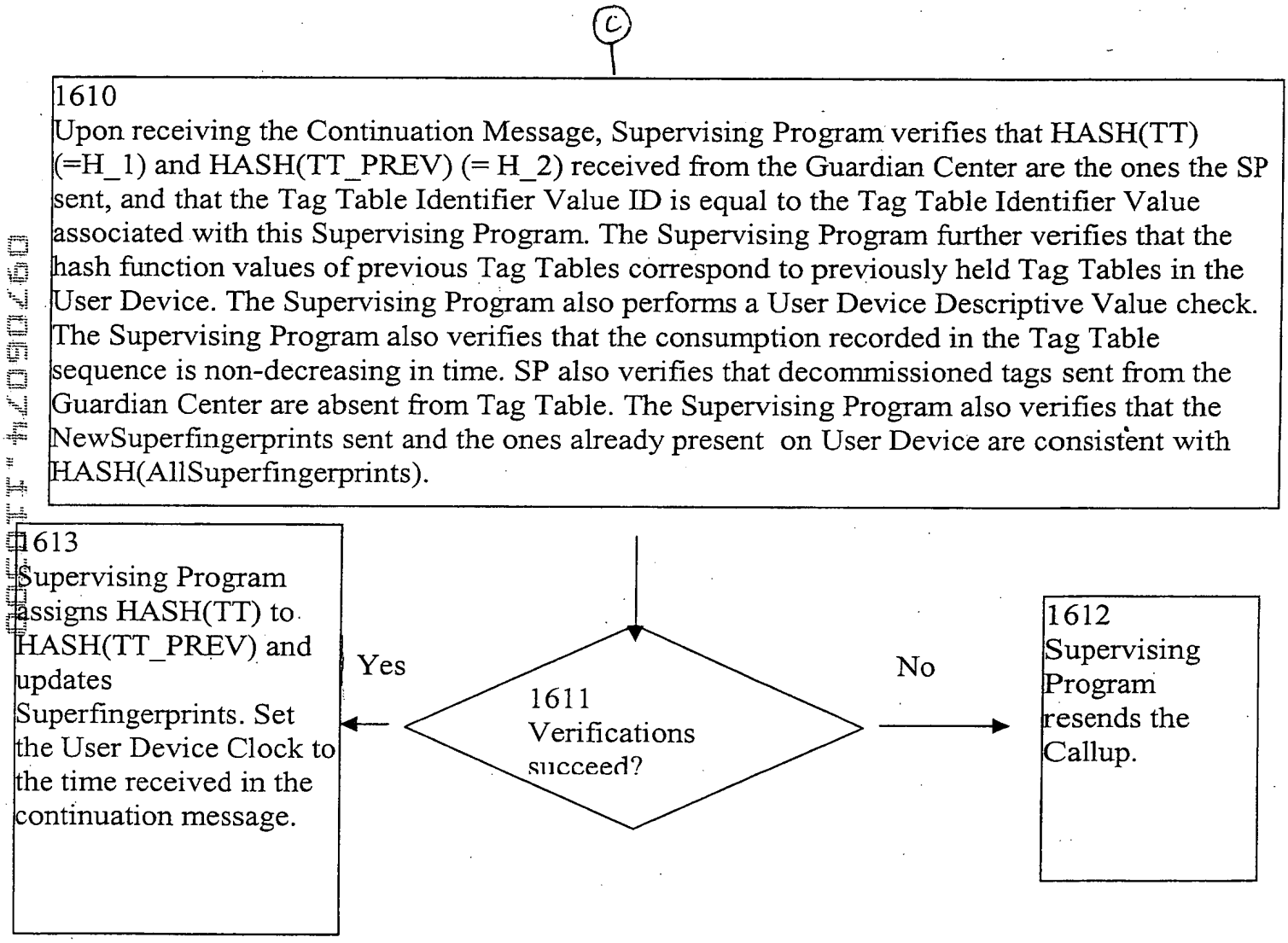


FIG. 13B

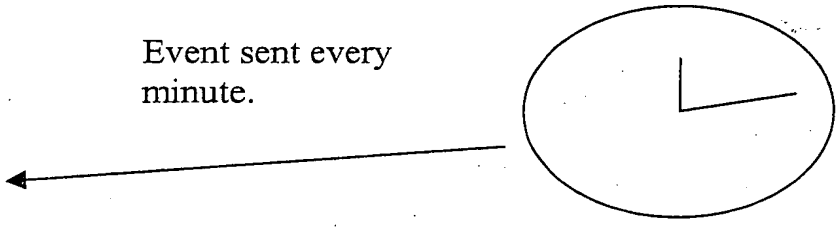


FIG. 14

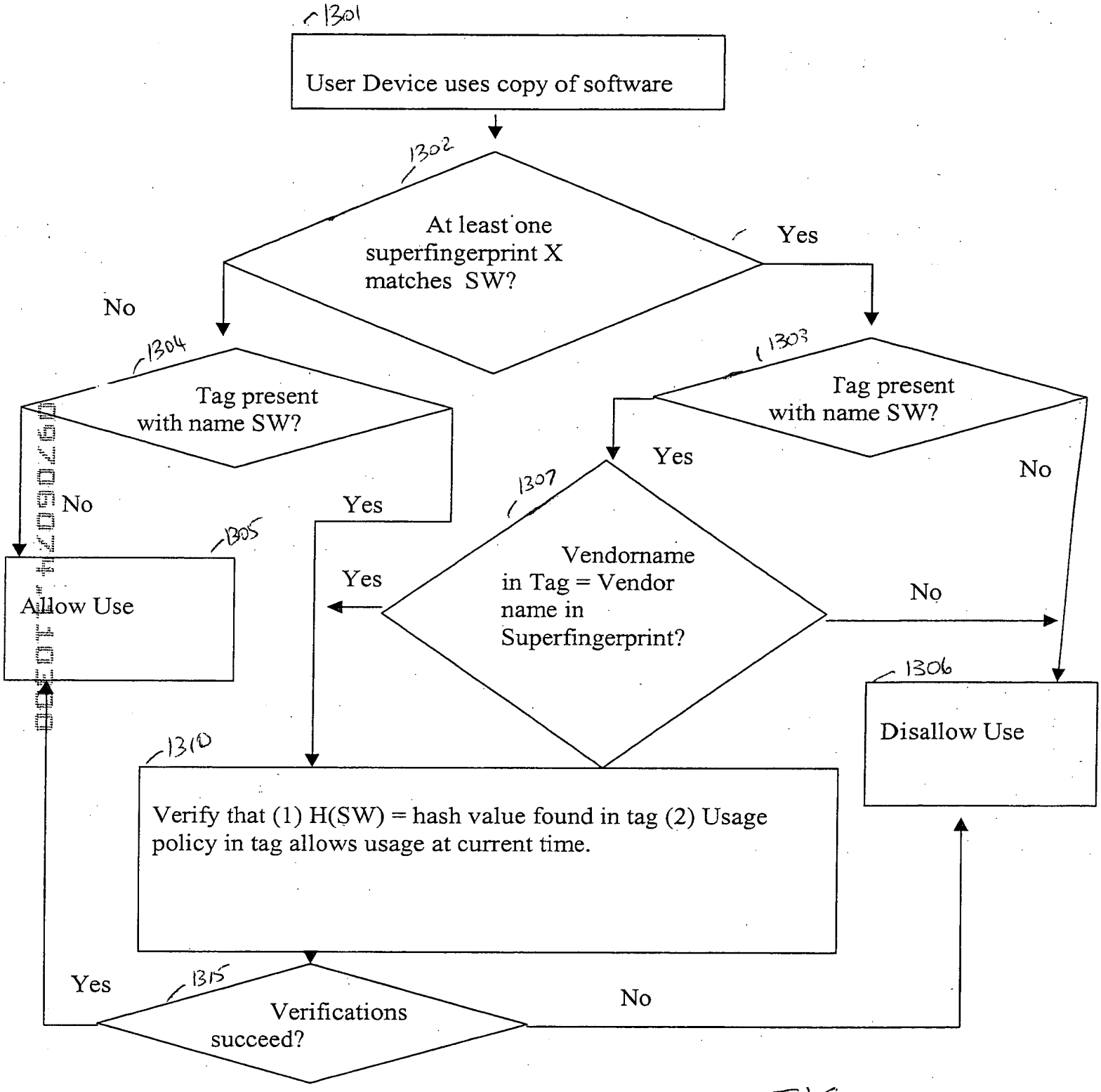

1301
User Device uses copy of software
1302
At least one superfingerprint X matches SW?
Yes
No
1303
Tag present with name SW?
Yes
No
1304
Tag present with name SW?
No
Yes
1305
Allow Use
1307
Vendorname in Tag = Vendor name in Superfingerprint?
Yes
No
1306
Disallow Use
1310
Verify that (1) H(SW) = hash value found in tag (2) Usage policy in tag allows usage at current time.
1315
Verifications succeed?
Yes
No

FIG. 15

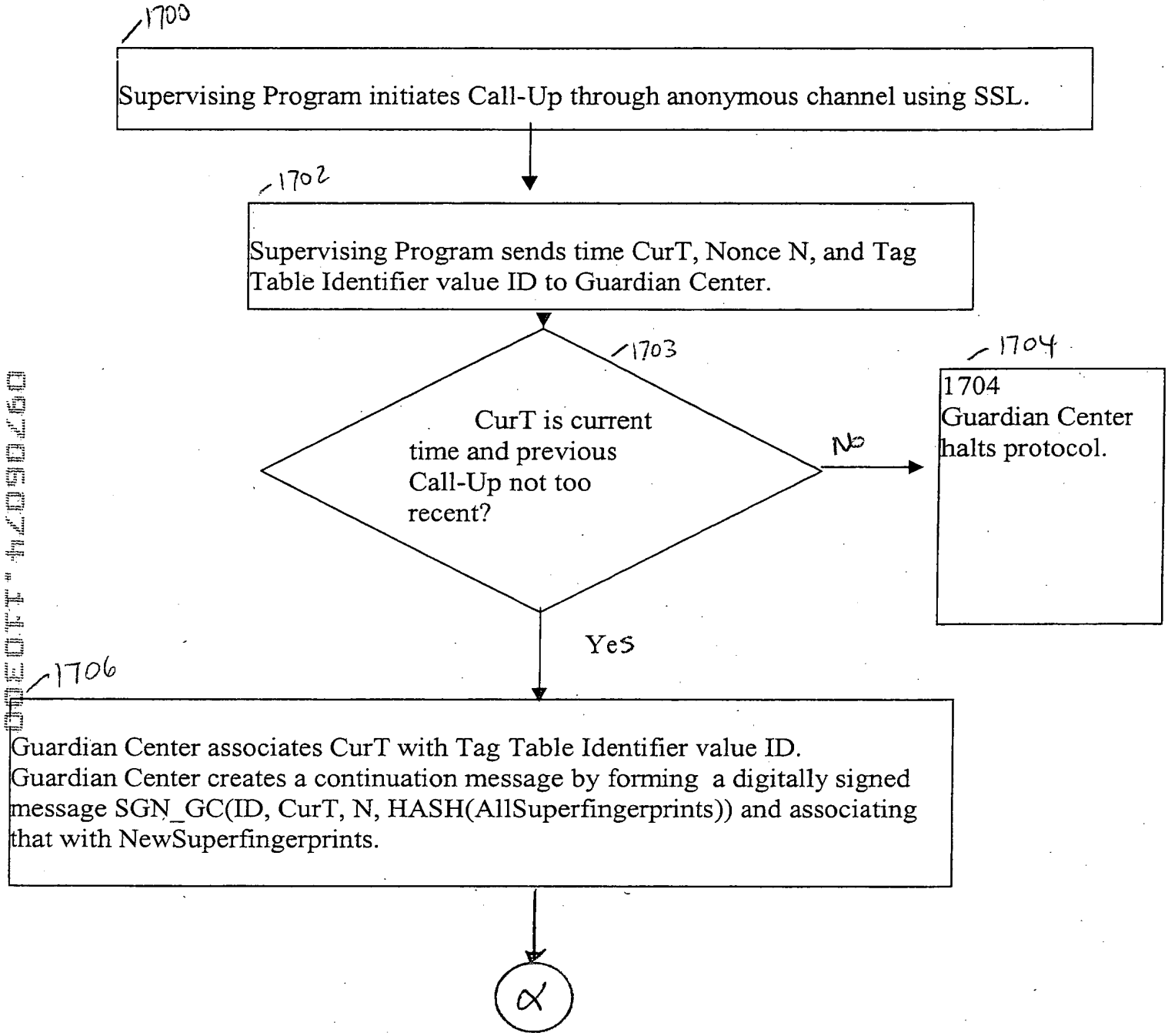

1700
Supervising Program initiates Call-Up through anonymous channel using SSL.
1702
Supervising Program sends time CurT, Nonce N, and Tag Table Identifier value ID to Guardian Center.
1703
CurT is current time and previous Call-Up not too recent?
No
1704
1704 Guardian Center halts protocol.
Yes
1706
Guardian Center associates CurT with Tag Table Identifier value ID. Guardian Center creates a continuation message by forming a digitally signed message SGN_GC(ID, CurT, N, HASH(AllSuperfingerprints)) and associating that with NewSuperfingerprints.
α

Fig 16 A

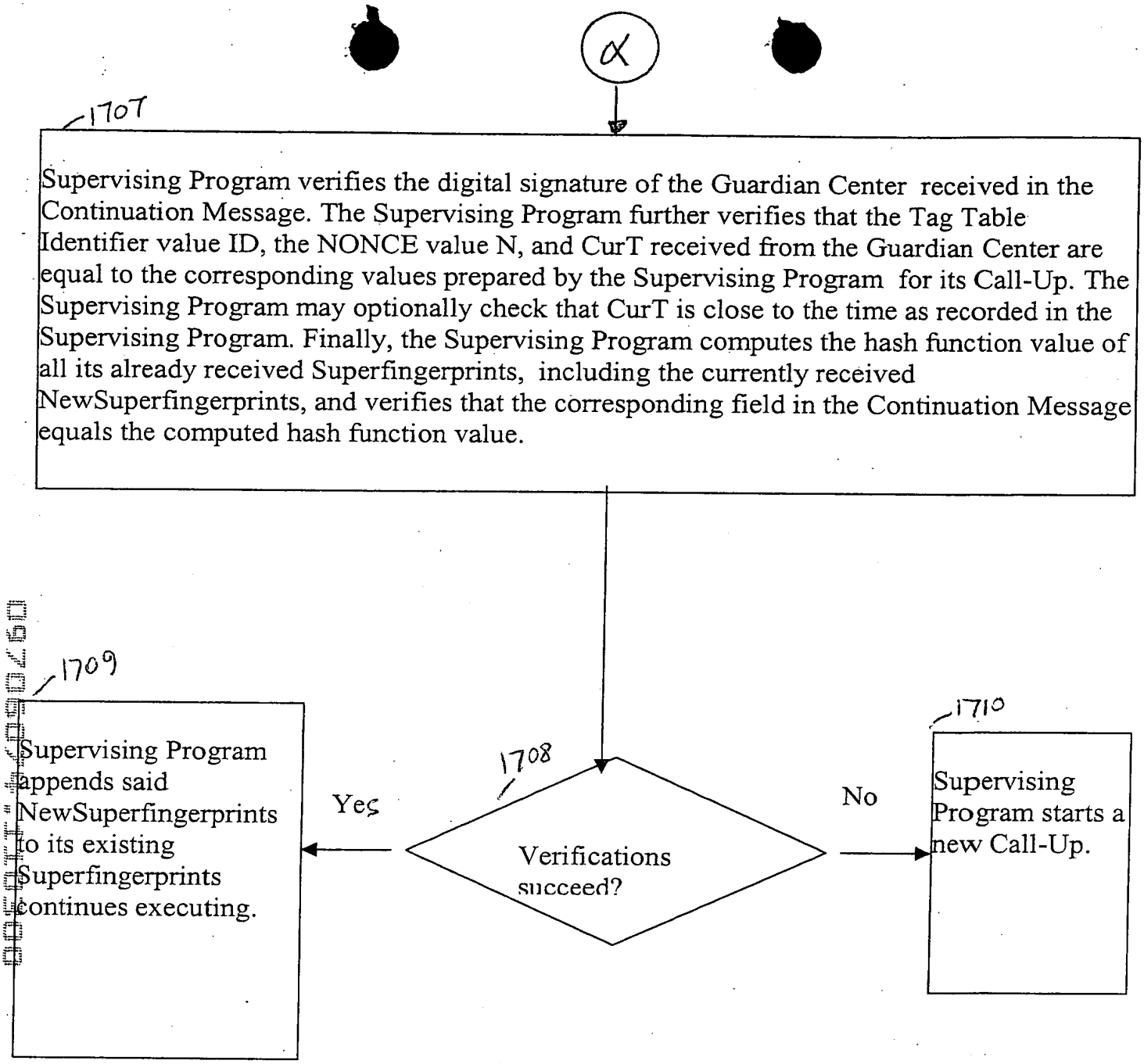

α
1707
Supervising Program verifies the digital signature of the Guardian Center received in the Continuation Message. The Supervising Program further verifies that the Tag Table Identifier value ID, the NONCE value N, and CurT received from the Guardian Center are equal to the corresponding values prepared by the Supervising Program for its Call-Up. The Supervising Program may optionally check that CurT is close to the time as recorded in the Supervising Program. Finally, the Supervising Program computes the hash function value of all its already received Superfingerprints, including the currently received NewSuperfingerprints, and verifies that the corresponding field in the Continuation Message equals the computed hash function value.
1708
Verifications succeed?
Yes
No
1709
Supervising Program appends said NewSuperfingerprints to its existing Superfingerprints continues executing.
1710
Supervising Program starts a new Call-Up.

FIG. 16B